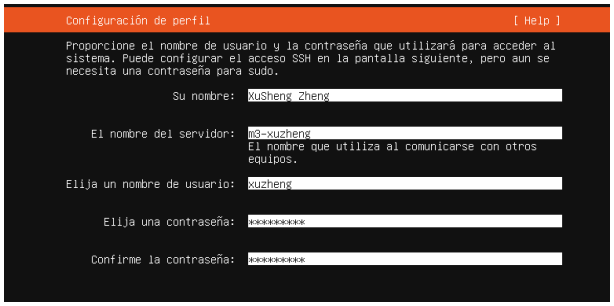
Práctica 3 SWAP

XuSheng Zheng

Índice

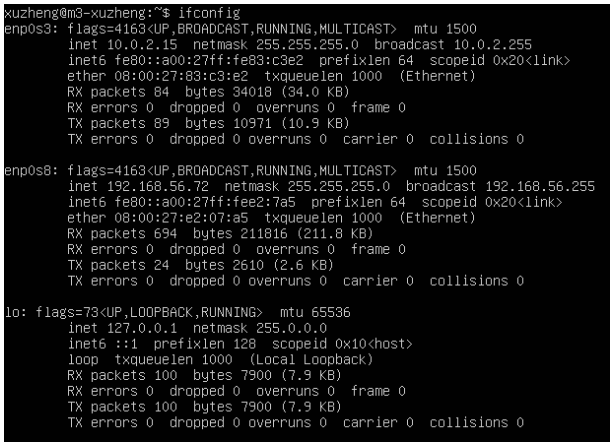
1. Creación de la tercera máquina

Para esta práctica, necesitamos crear una tercera máquina virtual que nos servirá de balanceador. Para ello, especificamos los siguientes datos siguiendo los mismos pasos que en la práctica 1:



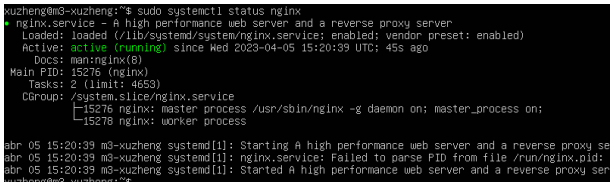
Una vez instalada la máquina, necesitamos configurar la conexión a red. Lo hacemos de la misma manera que en la práctica 1:

Y comprobamos con **ifconfig**:



2. Nginx

Instalamos **nginx** en m3 mediante **apt-get** con los comandos del guion y comprobamos que está activado:



Ahora procedemos a la configuración: en primer lugar deshabilitamos la funcionalidad de servidor web editando en */etc/nginx/nginx.conf* comentando la línea *include /etc/nginx/sites-enabled/*;* :

```
        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        #include /etc/nginx/sites-enabled/*;
```

Creamos el archivo de configuración */etc/nginx/conf.d/default.conf* con los siguientes datos:

```
upstream balanceo_xuzheng{
        server 192.168.56.70;
        server 192.168.56.71;
}

server{
        listen 80;
        server_name balanceador_xuzheng;

        access_log /var/log/nginx/balanceador_xuzheng.access.log;
        error_log /var/log/nginx/balanceador_xuzheng.error.log;
        root /var/www/;

        location /
        {
                proxy_pass http://balanceo_xuzheng;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";

        }
}
```

Una vez guardado, reiniciamos el servicio **nginx**. Para comprobar el funcionamiento del balanceador, se ha modificado el archivo */var/www/html/swap.html* para que se puedan distinguir las máquinas:

```
xuzheng@m1-xuzheng:~$ cat /var/www/html/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>
```

```
xuzheng@m2-xuzheng:~$ cat /var/www/html/swap.html
<HTML>
<BODY>
SWAP M2
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>
```

Ahora podemos comprobar el funcionamiento de **nginx** usando **cURL** desde el anfitrión:

```
txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M2
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>
```

En este caso hemos tratado las dos máquinas por iguales. Podemos dar más peso a la primera máquina mediante el modificador **weight**:

```
upstream balanceo_xuzheng{
        server 192.168.56.70 weight=2;
        server 192.168.56.71 weight=1;
}

server{
        listen 80;
        server_name balanceador_xuzheng;

        access_log /var/log/nginx/balanceador_xuzheng.access.log;
        error_log /var/log/nginx/balanceador_xuzheng.error.log;
        root /var/www/;

        location /
        {
                proxy_pass http://balanceo_xuzheng;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";

        }

}
```

Ahora de cada 3 peticiones, 2 serán atendidas por m1 y 1 por m2:

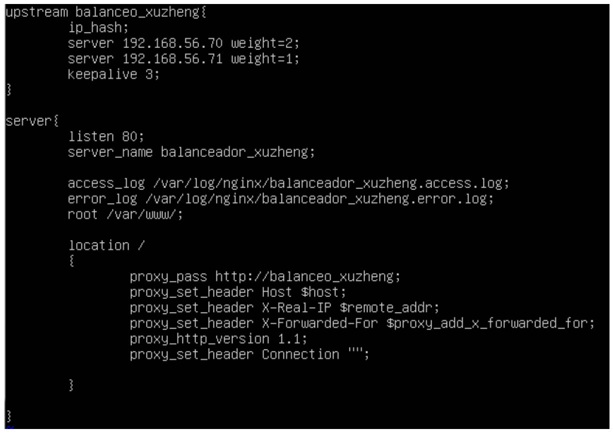
```
txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M2
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>
```

2.1. Opciones avanzadas

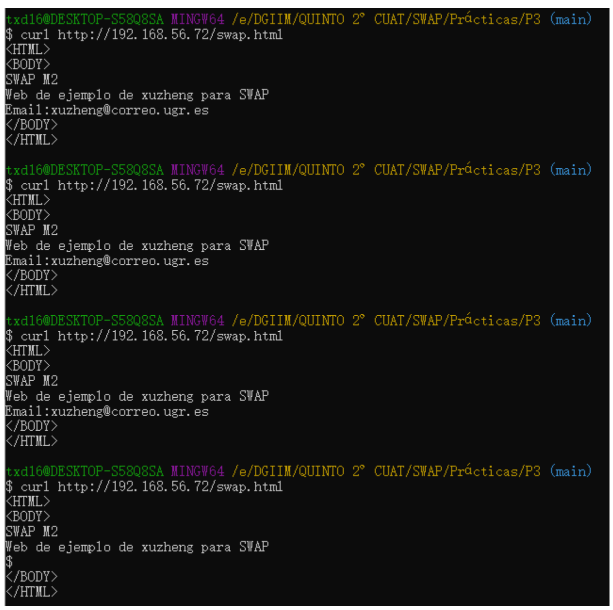
Para evitar conflictos de estados y sesiones entre los servidores, **nginx** permite dirigir las peticiones provenientes de un determinado IP al mismo servidor final mediante la opción **ip_hash**.

La directiva **keepalive** determina el número máximo de conexiones a servidores finales que se mantiene en caché, mientras que **keepalive_requests** determina el número máximo de peticiones que se pueden

servir a través de estas conexiones. Una vez que el número de peticiones excede el máximo, la conexión se cerrará.



Comprobamos el funcionamiento a través del anfitrión:



Podemos que en este caso sólo sirve m2 pues estamos enviando a través del mismo IP.

Para gestionar caídas de los servidores podemos utilizar las directivas **max_fails** para especificar el número máximo de intentos de comunicación fallidos antes de considerar al servidor no operativo, y **fail_timeout** para especificar el periodo de tiempo en el que se considera esos intentos. En el siguiente ejemplo, consideramos que el servidor no será operativo si existen 3 intentos fallidos en un periodo de 30 segundos:

```
upstream balanceo_xuzheng{
        ip_hash;
        server 192.168.56.70 weight=2 max_fails=3 fail_timeout=30s;
        server 192.168.56.71 weight=1 max_fails=3 fail_timeout=30s;
        keepalive 3;
}

server{
        listen 80;
        server_name balanceador_xuzheng;

        access_log /var/log/nginx/balanceador_xuzheng.access.log;
        error_log /var/log/nginx/balanceador_xuzheng.error.log;
        root /var/www/;

        location /
        {
                proxy_pass http://balanceo_xuzheng;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";

        }
}
```

3. Haproxy

Para este apartado, paramos y desactivamos **nginx** para evitar conflictos e instalamos **haproxy** mediante **apt-get**.

```
xuzheng@m3-xuzheng:~$ sudo systemctl disable nginx.service
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install
.
Executing: /lib/systemd/systemd-sysv-install disable nginx
xuzheng@m3-xuzheng:~$ sudo systemctl stop nginx.service
xuzheng@m3-xuzheng:~$ sudo systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:nginx(8)

abr 07 09:02:08 m3-xuzheng systemd[1]: Starting A high performance web server and a reverse proxy se
abr 07 09:02:08 m3-xuzheng systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid:
abr 07 09:02:08 m3-xuzheng systemd[1]: Started A high performance web server and a reverse proxy ser
abr 07 09:12:49 m3-xuzheng systemd[1]: Stopping A high performance web server and a reverse proxy se
abr 07 09:12:49 m3-xuzheng systemd[1]: Stopped A high performance web server and a reverse proxy ser
abr 07 09:12:49 m3-xuzheng systemd[1]: Starting A high performance web server and a reverse proxy se
abr 07 09:12:49 m3-xuzheng systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid:
abr 07 09:12:49 m3-xuzheng systemd[1]: Started A high performance web server and a reverse proxy ser
abr 07 09:16:20 m3-xuzheng systemd[1]: Stopping A high performance web server and a reverse proxy se
abr 07 09:16:20 m3-xuzheng systemd[1]: Stopped A high performance web server and a reverse proxy ser
xuzheng@m3-xuzheng:~$ sudo apt-get install haproxy
```

Comprobamos que está activo:

```
xuzheng@m3-xuzheng:~$ sudo systemctl status haproxy.service
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2023-04-07 09:17:36 UTC; 1min 25s ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
 Main PID: 2720 (haproxy)
    Tasks: 2 (limit: 4653)
   CGroup: /system.slice/haproxy.service
           ├─2720 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
           └─2721 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid

abr 07 09:17:36 m3-xuzheng systemd[1]: Starting HAProxy Load Balancer...
abr 07 09:17:36 m3-xuzheng systemd[1]: Started HAProxy Load Balancer.
```

Pasamos ahora a configurar **haproxy**: añadimos al archivo */etc/haproxy/haproxy.cfg* las siguientes líneas para tener una configuración round-robin básica:

```
frontend http-in
        bind *:80
        default_backend balanceo_xuzheng

backend balanceo_xuzheng
        server m1 192.168.56.70:80 maxconn 32
        server m2 192.168.56.71:80 maxconn 32
```

Reiniciamos el servicio y comprobamos desde el anfitrión:

Para dar más peso a determinados servidores, podemos usar la opción **weight** con números entre 0 y 256 en las líneas de **server**. Para establecer una analogía con el apartado de **nginx**, asignamos a m1 el doble de carga que m2:

```
frontend http-in
        bind *:80
        default_backend balanceo_xuzheng

backend balanceo_xuzheng
        server m1 192.168.56.70:80 weight 20 maxconn 32
        server m2 192.168.56.71:80 weight 10 maxconn 32
```

Y comprobamos:

Vemos que efectivamente m1 atiende 2 de las 3 peticiones.

3.1. Opciones avanzadas

Al igual que **nginx**, **haproxy** también permite el balanceo por IP mediante la opción **hash-type consistent**.

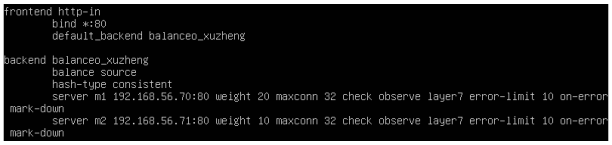
Además, **haproxy** permite 3 tipos de comprobaciones sobre el estado de los servidores:

- Activo: por defecto, en este caso, **haproxy** intentará establecer una conexión TCP con el servidor final cada 2 segundos. Tras 3 conexiones fallidas, se considerará que el servidor está caído y no se le enviarán peticiones hasta que consiga 2 conexiones exitosas.

- Pasivo: similar a la opción **keepalive_requests** de **nginx**, establece un límite de errores consecutivos que puede haber antes de dar por caído al servidor.

- Agente: mediante un agente externo del servidor final, **haproxy** puede controlar el estado con el que se encuentra el servidor.

En el siguiente ejemplo establecemos un máximo de 10 errores:



Con **observe layer7** indicamos que se consideren todas las respuestas HTTP del servidor y con **on-error mark-down** indicamos que se considere el servidor caído cuando se alcanzan los 10 errores. Para comprobar el funcionamiento del balanceo por IP, mandamos peticiones desde el anfitrión:

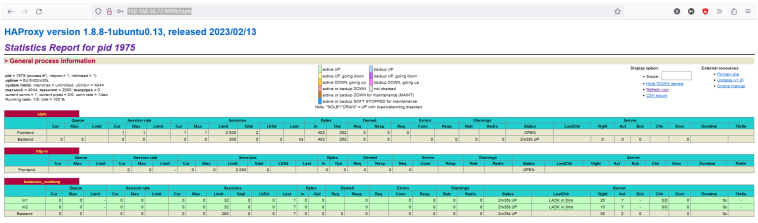
Podemos apreciar que todas las peticiones son atendidas por m1 pues provienen del mismo IP.

4. Módulo de estadísticas

Para habilitar el módulo de estadísticas de Haproxy, añadimos las siguientes líneas al archivo */etc/haproxy /haproxy.cfg*:

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # Default ciphers to use on SSL-enabled listening sockets.
        # For more information, see ciphers(1SSL). This list is from:
        #  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
        # An alternative list with additional directives can be obtained from
        #  https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy
        ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+
AESGCM:RSA+AES:!aNULL:!MD5:!DSS
        ssl-default-bind-options no-sslv3


        #Lineas añadidas:
        stats socket /var/lib/haproxy/stats

listen stats
        bind *:9999
        mode http
        stats enable
        stats uri /stats
        stats realm HAProxy\ Statistics
        stats auth xuzheng:xuzheng
```

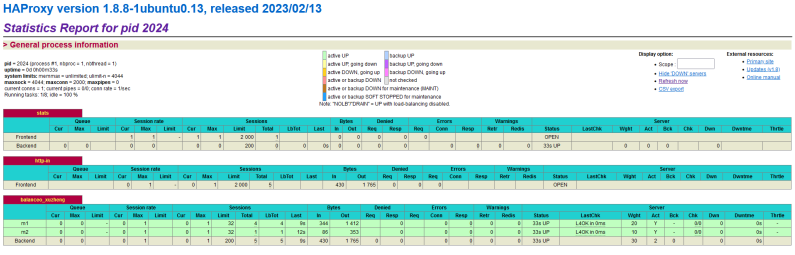
Reiniciamos el servicio y ahora podemos acceder a través de la url *http://192.168.56.72:9999/stats* con el usuario y la contraseña que hemos configurado:



Para poder testear el módulo de estadísticas, deshabilitamos de momento el balanceo por IP:

```
frontend http-in
        bind *:80
        default_backend balanceo_xuzheng

backend balanceo_xuzheng
        server m1 192.168.56.70:80 weight 20 maxconn 32 check observe layer7 error-limit 10 on-error
 mark-down
        server m2 192.168.56.71:80 weight 10 maxconn 32 check observe layer7 error-limit 10 on-error
 mark-down
```

Tras reiniciar el servicio lanzamos varias peticiones al balanceador y observamos que los datos recogidos por el módulo son correctos:



En particular se verifica que m1 haya servido 4 de las 5 peticiones.

4.1. Opciones avanzadas

Para que la página de estadísticas se actualice automáticamente podemos usar **stats refresh** con el periodo de tiempo con el que queremos que se actualice. Para limitar el número de conexiones a la página podemos usar **stats maxconn** y se puede establecer un timeout mediante **stats timeout**.

En el siguiente ejemplo, establecemos un refresco de página cada 10 segundos, un timeout de 30 segundos y un número máximo de 5 conexiones:

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
        stats timeout 30s
        stats maxconn 5
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # Default ciphers to use on SSL-enabled listening sockets.
        # For more information, see ciphers(1SSL). This list is from:
        #  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
        # An alternative list with additional directives can be obtained from
        #  https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy
        ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+
AESGCM:RSA+AES:!aNULL:!MD5:!DSS
        ssl-default-bind-options no-sslv3


        #Lineas añadidas:
        stats socket /var/lib/haproxy/stats

listen stats
        bind *:9999
        mode http
        stats enable
        stats uri /stats
        stats realm HAProxy\ Statistics
        stats auth xuzheng:xuzheng
        stats refresh 10s
```

5. Go-Between

Para evitar conflictos, deshabilitamos los balanceadores anteriores e instalamos **go-between** mediante **sudo snap install gobetween –edge**. Una vez instalado comprobamos el estado del servicio:

```
xuzheng@m3-xuzheng:~$ snap services gobetween
Service              Startup  Current  Notes
gobetween.gobetween  enabled  active   -
```

Para configurar el algoritmo round-robin editamos el archivo */var/snap/gobetween/common/gobetween.toml* como sigue:

```
[servers.sample]
protocol = "tcp"
bind = "192.168.56.72:80"
balance = "roundrobin"

  [servers.sample.discovery]
  kind = "static"
  static_list = [
      "192.168.56.70:80",
      "192.168.56.71:80"
  ]
```

Además, para evitar errores, se ha desactivado el servidor de API REST y el servidor de ejemplo para UDP:

```
# ---------- udp example ----------- #

#[servers.udpsample]
#bind = "localhost:4000"
#protocol = "udp"

#  [servers.udpsample.udp]
#  max_responses = 1

#  [servers.udpsample.discovery]
#  kind = "static"
#  static_list = [
#      "8.8.8.8:53",
#      "8.8.4.4:53",
#      "91.239.100.100:53"
#  ]
#
```

```
#
# REST API server configuration
#
[api]
enabled = false  # true | false
bind = ":8888"  # "host:port"
cors = false    # cross-origin resource sharing
```

Para lanzar el servicio ejecutamos lo siguiente:

```
xuzheng@m3-xuzheng:~$ sudo /snap/gobetween/current/bin/gobetween from-file /var/snap/gobetween/commo
n/gobetween.toml
gobetween v0.8.0+snapshot
2023-04-08 16:49:13 [INFO ] (manager): Initializing...
2023-04-08 16:49:13 [INFO ] (server): Creating 'sample': 192.168.56.72:80 roundrobin static none
2023-04-08 16:49:13 [INFO ] (scheduler): Starting scheduler sample
2023-04-08 16:49:13 [INFO ] (manager): Initialized
2023-04-08 16:49:13 [INFO ] (metrics): Metrics disabled
2023-04-08 16:49:13 [INFO ] (api): API disabled
```

Y comprobamos desde el anfitrión:

```
txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M2
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M2
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>
```

5.1. Opciones avanzadas

go-between también permite el balanceo por IP mediante la opción **balance=“iphash”** así como ponderación mediante **balance=“weight”**. Para seguir el mismo esquema de los anteriores balanceadores damos el doble de carga para m1:

```
# ---------- tcp example ----------- #

[servers.sample]
protocol = "tcp"
bind = "192.168.56.72:80"
balance = "weight"

  [servers.sample.discovery]
  kind = "static"
  static_list = [
      "192.168.56.70:80 weight=2",
      "192.168.56.71:80 weight=1"
  ]
```

Lanzamos el servicio y comprobamos desde el anfitrión:

```
txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M2
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>
```

A diferencia de los anteriores balanceadores, que pueden usar a la vez ponderación y round-robin, en este caso los pesos determinan la probabilidad de ser elegida para servir una petición. Lo que implica que no tenemos asegurado que de las 3 peticiones, 2 vayan a ser servidas por m1. Para controlar el estado de los servidores finales, **go-between** hace uso del apartado **healthcheck**. Veamos con el siguiente ejemplo:

```
[servers.sample]
protocol = "tcp"
bind = "192.168.56.72:80"
balance = "weight"

  [servers.sample.discovery]
  kind = "static"
  static_list = [
      "192.168.56.70:80 weight=2",
      "192.168.56.71:80 weight=1"
  ]

  [servers.sample.healthcheck]
  fails = 3
  passes = 2
  interval = "5s"
  kind = "ping"
  ping_timeout_duration = "500ms"
```

Con esta configuración estamos indicando que se realice un chequeo sobre el estado de los servidores por

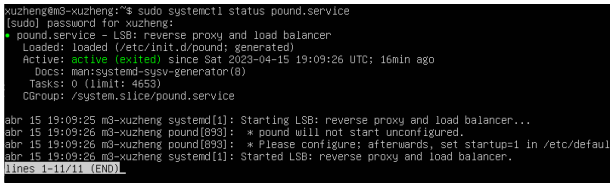
cada 5 segundos mediante conexiones por **ping**. Si una conexión no se establece en 500 ms se considera una pérdida. Tras 3 pérdidas, se considerará que el servidor esta caído. Mientras que 2 conexiones exitosas a un servidor caído se considerará que se ha recuperado.

Para evitar conflicto con el siguiente balanceador, deshabilitamos **gobetween** con el comando **sudo snap disable gobetween**.

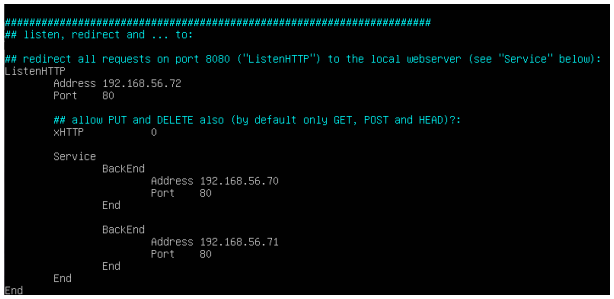
6. Pound

Actualmente no se puede instalar **pound** mediante **apt-get**, pero se ha encontrado un repositorio que almacena archivos para la instalación de **pound**. Dado que la versión de nuestro Ubuntu es la 18, la versión de **pound** que nos funciona es la 2.6.

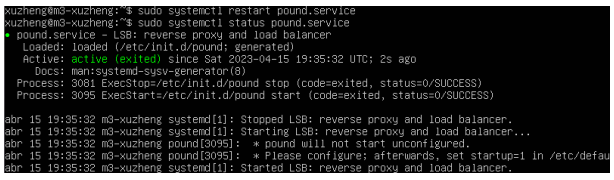
Por tanto, descargamos el archivo *pound_2.6-2_amd64.deb* de `http://old.kali.org/kali/pool/main/p/pound/` con el comando **curl -O http://old.kali.org/kali/pool/main/p/pound/pound_2.6-2_amd64.deb** e instalamos el paquete con **sudo dpkg -i pound_2.6-2_amd64.deb**. Una vez instalado podemos comprobar su estado:



Para configurar, editamos el archivo */etc/pound/pound.cfg*:



Al reiniciar el servicio nos pide que añadamos **startup=1** al documento */etc/default/pound*:



Lo añadimos y reiniciamos el servicio.

```
# Defaults for pound initscript
# sourced by /etc/init.d/pound
# installed at /etc/default/pound by the maintainer scripts

# prevent startup with default configuration
# set the below varible to 1 in order to allow pound to start
startup=1
```

```
xuzheng@m3-xuzheng:~$ sudo systemctl restart pound.service
xuzheng@m3-xuzheng:~$ sudo systemctl status pound.service
● pound.service - LSB: reverse proxy and load balancer
     Loaded: loaded (/etc/init.d/pound; generated)
     Active: active (running) since Sat 2023-04-15 19:42:29 UTC; 2s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 3437 ExecStop=/etc/init.d/pound stop (code=exited, status=0/SUCCESS)
    Process: 3458 ExecStart=/etc/init.d/pound start (code=exited, status=0/SUCCESS)
      Tasks: 132 (limit: 4653)
     CGroup: /system.slice/pound.service
             ├─3464 /usr/sbin/pound
             └─3465 /usr/sbin/pound

abr 15 19:42:29 m3-xuzheng systemd[1]: Stopped LSB: reverse proxy and load balancer.
abr 15 19:42:29 m3-xuzheng systemd[1]: Starting LSB: reverse proxy and load balancer...
abr 15 19:42:29 m3-xuzheng pound[3458]:  * Starting reverse proxy and load balancer pound
abr 15 19:42:29 m3-xuzheng pound[3458]: starting...
abr 15 19:42:29 m3-xuzheng pound[3458]:    ...done.
abr 15 19:42:29 m3-xuzheng systemd[1]: Started LSB: reverse proxy and load balancer.
```

Comprobamos que funciona correctamente desde el anfitrión:

```
txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2° CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2° CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M2
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>
```

También podemos otorgar más prioridad para una máquina que otra:

```
ListenHTTP
        Address 192.168.56.72
        Port    80

        ## allow PUT and DELETE also (by default only GET, POST and HEAD)?:
        xHTTP           0

        Service
                BackEnd
                        Address 192.168.56.70
                        Port    80
                        Priority 2
                End

                BackEnd
                        Address 192.168.56.71
                        Port    80
                        Priority 1
                End
        End
End
```

6.1. Opciones avanzadas

Para establecer el balanceo por IP podemos utilizar el bloque **session**, éste nos permite configurar el tiempo máximo en el que se mantiene una sesión, en nuestro caso se ha decidido un margen de 20 minutos:

```
ListenHTTP
        Address 192.168.56.72
        Port    80

        ## allow PUT and DELETE also (by default only GET, POST and HEAD)?:
        xHTTP           0

        Service
                BackEnd
                        Address 192.168.56.70
                        Port    80
                        Priority 2
                End

                BackEnd
                        Address 192.168.56.71
                        Port    80
                        Priority 1
                End

                Session
                        Type IP
                        TTL  1200
                End
        End
End
```

Comprobamos desde el anfitrión:

```
txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>

txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ curl http://192.168.56.72/swap.html
<HTML>
<BODY>
SWAP M1
Web de ejemplo de xuzheng para SWAP
Email:xuzheng@correo.ugr.es
</BODY>
</HTML>
```

Para controlar el estado de los servidores, se puede configurar las directivas globales **TimeOut** y **ConnTO** que establecen el tiempo que espera **Pound** para una respuesta del servidor y una conexión al servidor, respectivamente. Estas directivas se pueden sobreescribir en cada servidor. Además, **Alive** determina cada cuánto tiempo **Pound** chequea el estado de los servidores. En nuestro caso, asignamos los siguientes valores:

```
## check backend every X secs:
Alive           20
TimeOut         10
ConnTO          10
```

7. Someter a carga la granja web con AB

A partir de esta sección, se ha eliminado la configuración respecto al balanceo por IP de todos los balanceadores.

En primer lugar instalamos **Apache Benchmark** en el anfitrión. Dado que en este caso el anfitrión tiene el sistema operativo Windows, se han seguido los pasos del videotutorial `https://www.youtube.com/watch?v=hUZso9TpEes`.

Una vez instalado probamos solicitar 10000 veces la página *swap.html* en peticiones concurrentes de 10 en 10:

```
ixdlb@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ ab -n 10000 -c 10 http://192.168.56.72/swap.html
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.72 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        Apache/2.4.41
Server Hostname:        192.168.56.72
Server Port:            80

Document Path:          /swap.html
Document Length:        102 bytes

Concurrency Level:      10
Time taken for tests:   50.059 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      3720000 bytes
HTML transferred:       1020000 bytes
Requests per second:    199.76 [#/sec] (mean)
Time per request:       50.059 [ms] (mean)
Time per request:       5.006 [ms] (mean, across all concurrent requests)
Transfer rate:          72.57 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1    5   1.1      5      20
Processing:    16   45   5.7     44     535
Waiting:       14   37   8.8     39     503
Total:         20   49   5.7     49     536

Percentage of the requests served within a certain time (ms)
  50%     49
  66%     50
  75%     50
  80%     50
  90%     51
  95%     52
  98%     54
  99%     56
 100%    536 (longest request)
```

Podemos ver que en este caso el tiempo medio por petición es de 50 ms.

7.1. Otras opciones

Aparte de las opciones **-n** y **-c** visto anteriormente, **ab** tiene otras opciones interesantes:

- **-e csv-file**: escribe en un csv el tiempo que se tardó para procesar cada porcentaje de las peticiones totales.
- **-q**: elimina los mensajes de progreso.
- **-p, -u file**: permite añadir fichero con datos para peticiones POST y PUT respectivamente. Es necesario usar también la opción **-T**.

- **-t timelimit**: determina el número máximo de segundos para el benchmark.

Como ejemplo utilizamos el siguiente comando:

```
txd16@DESKTOP-S58Q8SA MINGW64 /e/DGIIM/QUINTO 2º CUAT/SWAP/Prácticas/P3 (main)
$ ab -n 1000 -c 10 -q -e ejemplo.csv  http://192.168.56.72/swap.html
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.72 (be patient).....done


Server Software:        Apache/2.4.41
Server Hostname:        192.168.56.72
Server Port:            80

Document Path:          /swap.html
Document Length:        102 bytes

Concurrency Level:      10
Time taken for tests:   5.978 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      372000 bytes
HTML transferred:       102000 bytes
Requests per second:    167.28 [#/sec] (mean)
Time per request:       59.781 [ms] (mean)
Time per request:       5.978 [ms] (mean, across all concurrent requests)
Transfer rate:          60.77 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1    5   1.3      5      17
Processing:    17   47  22.8     46     537
Waiting:       12   36  18.8     35     504
Total:         21   52  22.7     51     538

Percentage of the requests served within a certain time (ms)
  50%     51
  66%     51
  75%     51
  80%     51
  90%     52
  95%     54
  98%     58
  99%     60
 100%    538 (longest request)
```

Podemos ver que en este caso ya no nos muestra el mensaje sobre el número de peticiones completadas y si abrimos *ejemplo.csv* tenemos lo siguiente (se muestra sólo parte del csv):

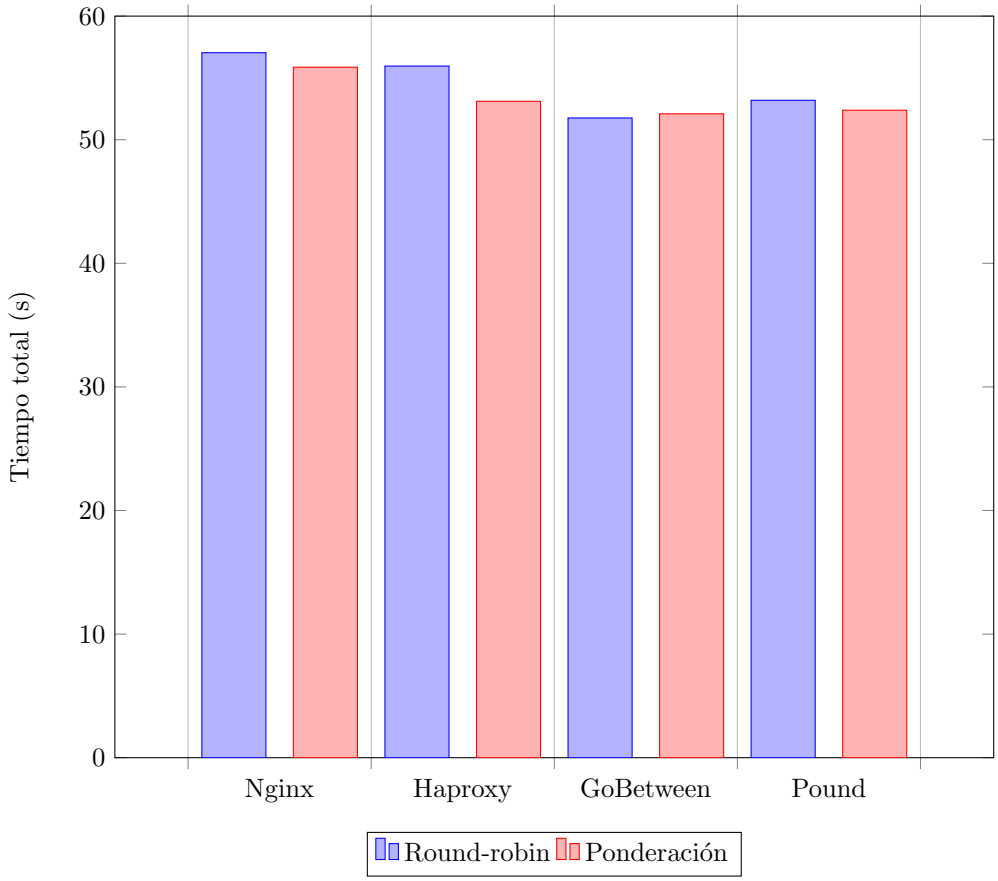
```
Percentage served,Time in ms
0,21.473
1,41.967
2,43.920
3,45.278
4,47.824
5,48.799
6,49.630
7,49.775
8,49.775
9,49.775
10,49.775
11,49.775
12,49.775
13,49.775
14,49.775
15,49.775
16,49.776
17,49.776
18,49.776
19,49.776
20,49.776
```

8. Análisis comparativo de los balanceadores

A continuación presentamos los resultados de los balanceadores sometiéndolos a una carga de 10000 peticiones concurrentes de 10 en 10. Para cada balanceador se testea tanto con round-robin como con ponderación con m1 el doble de carga que m2.

Balanceador	Algoritmo	Tiempo total (s)	Peticiones/s	Longest request (ms)
Nginx	Round-robin	57.038	175.32	563
Nginx	Ponderación	55.867	179.00	542
Haproxy	Round-robin	55.960	178.70	541
Haproxy	Ponderación	53.108	188.30	538
GoBetween	Round-robin	51.757	193.21	545
GoBetween	Ponderación	52.094	191.96	552
Pound	Round-robin	53.190	188.01	543
Pound	Ponderación	52.388	190.88	538

Puesto que el número de peticiones por segundo se puede obtener a partir del tiempo total, basta con analizar esta última variable. Con los anteriores datos podemos generar el siguientes diagrama de barras:



Podemos ver que en nuestro caso, **GoBetween** es el más rápido para ambos algoritmos, con casi 7 segundos de diferencia respecto de **Nginx** en el caso de usar round-robin.

Cabe destacar que en el caso de **Nginx** y **Haproxy** el tiempo con round-robin es superior que el tiempo con ponderación, y en el caso de **GoBetween** y **Pound** ocurre justo lo contrario. Esto puede deberse a múltiples factores: retardos debido al sistema operativo del anfitrión, el funcionamiento del propio balanceador u otros

motivos desconocidos. Pero lo lógico sería que round-robin funcionara mejor que ponderación, ya que en el caso de ponderación estamos cargando más a m1 cuando ambas máquinas son iguales.

Finalmente, cabe destacar que este análisis no determina la superioridad de un balanceador respecto a los otros, pues se ha analizado solamente datos de un único benchmark para los dos algoritmos. Pero a partir de los resultados, vemos que en nuestro caso, **GoBetween** tiene el mejor rendimiento y **Nginx** el peor.

9. Bibliografía

- http://nginx.org/en/docs/http/ngx_http_upstream_module.html
- http://www.haproxy.org/download/1.4/doc/configuration.txt
- https://serverfault.com/questions/113637/haproxy-roundrobin-weights
- https://www.haproxy.com/blog/client-ip-persistence-or-source-ip-hash-load-balancing
- https://www.haproxy.com/blog/how-to-enable-health-checks-in-haproxy
- https://www.haproxy.com/blog/exploring-the-haproxy-stats-page/
- https://gobetween.io/documentation.html#Static-balancing
- http://old.kali.org/kali/pool/main/p/pound/
- https://www.linuxhelp.com/how-to-configure-load-balancer-with-pound-in-ubuntu
- https://linux.die.net/man/8/pound
- https://www.apachelounge.com/download/
- https://www.youtube.com/watch?v=hUZso9TpEes
- https://httpd.apache.org/docs/2.2/programs/ab.html